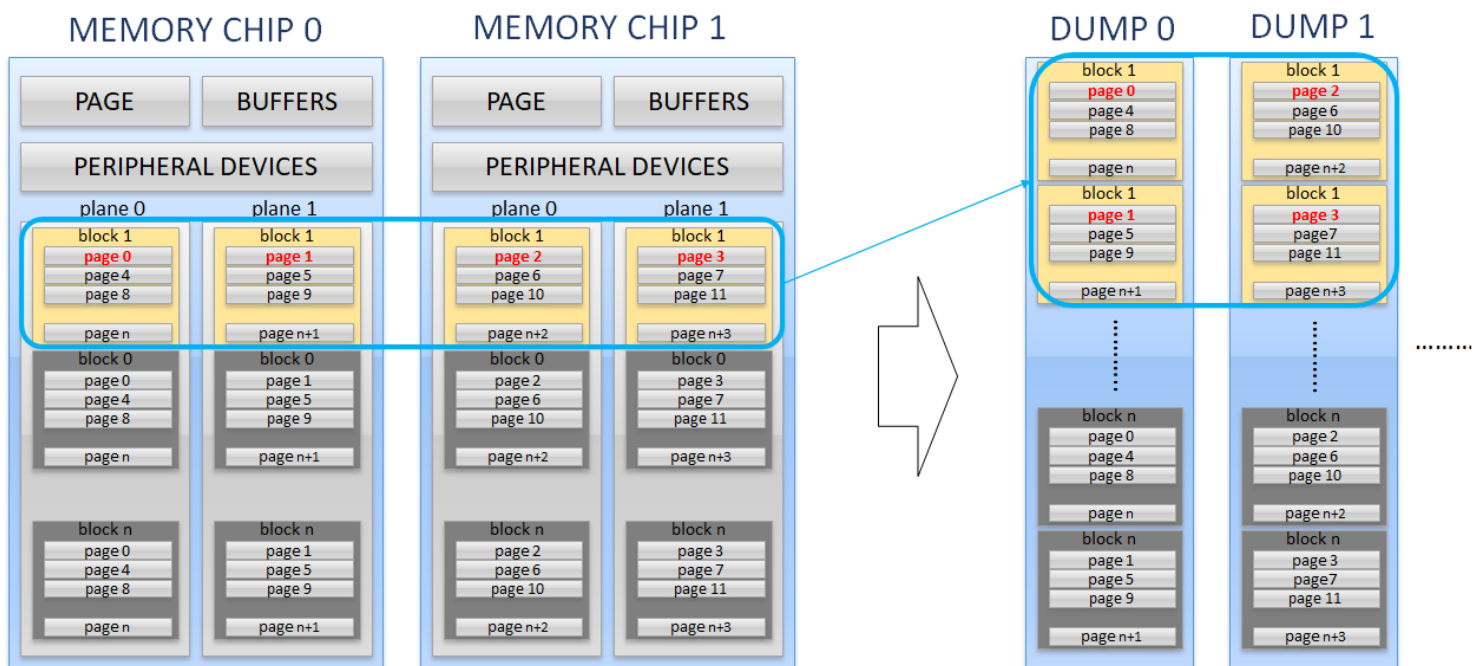


Asymmetric Multi-Plane page allocation

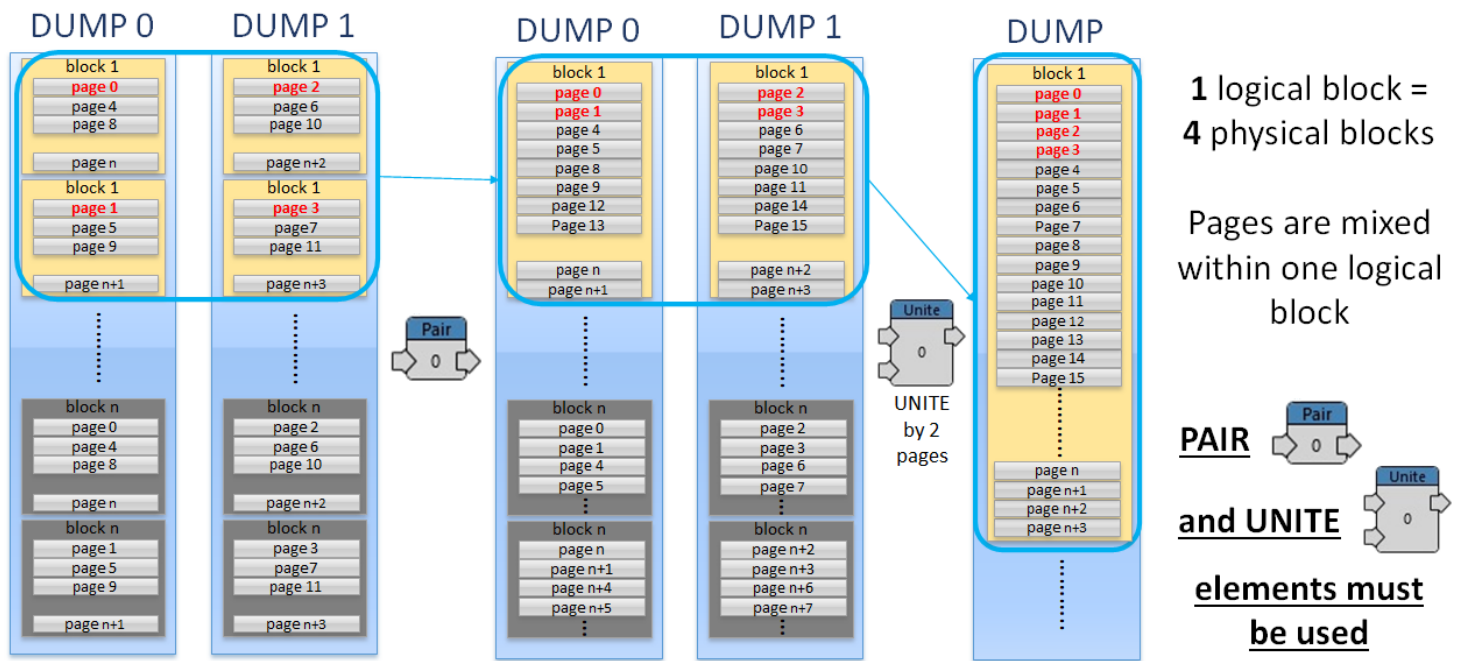
Standard Multi-Plane page allocation

To understand how the asymmetric multi-plane page allocation works it is necessary to know how the standard one works. This is an important topic because many devices are using it, and almost all of them are working in the standard algorithm.

MULTI-PLANE PAGE ALLOCATION ON THE CHIP LEVEL



MULTI-PLANE PAGE ALLOCATION ON THE CHIP LEVEL

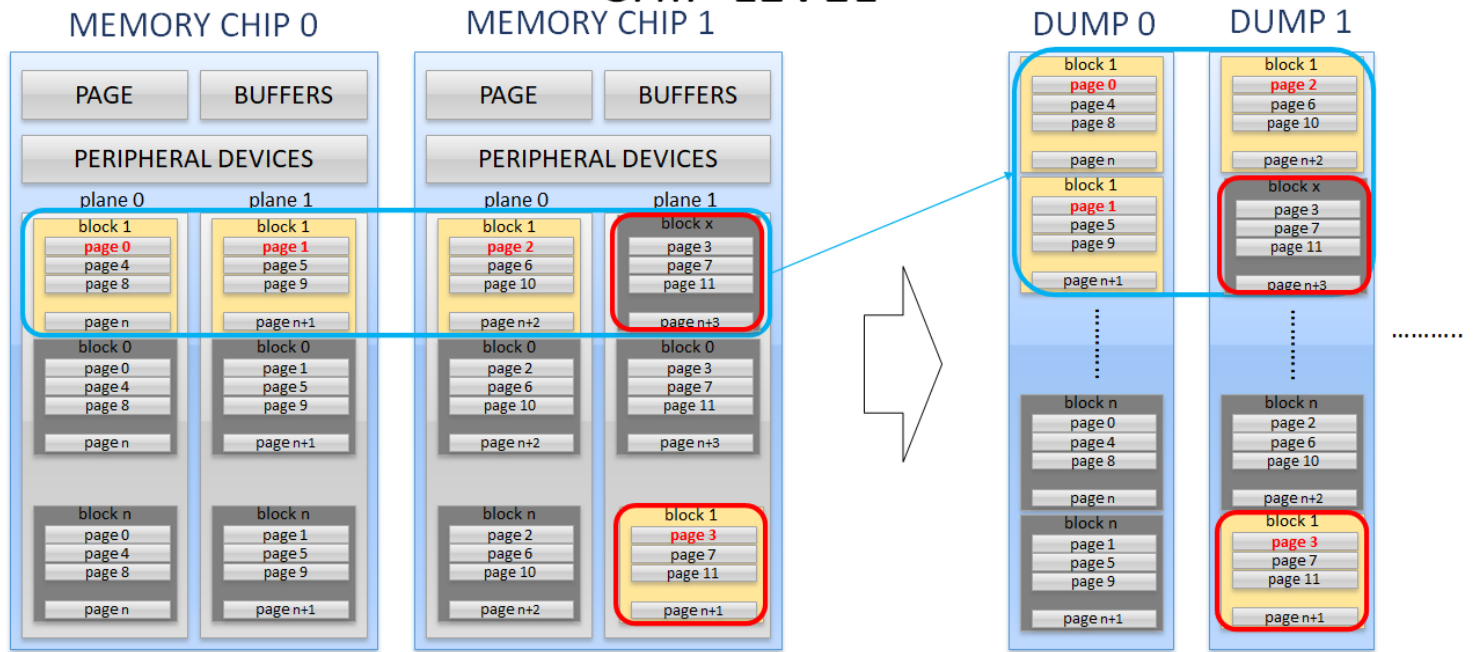


[More about standard Multi-Plane Allocation here.](https://support.rusolut.com/portal/en/kb/articles/doc1p)

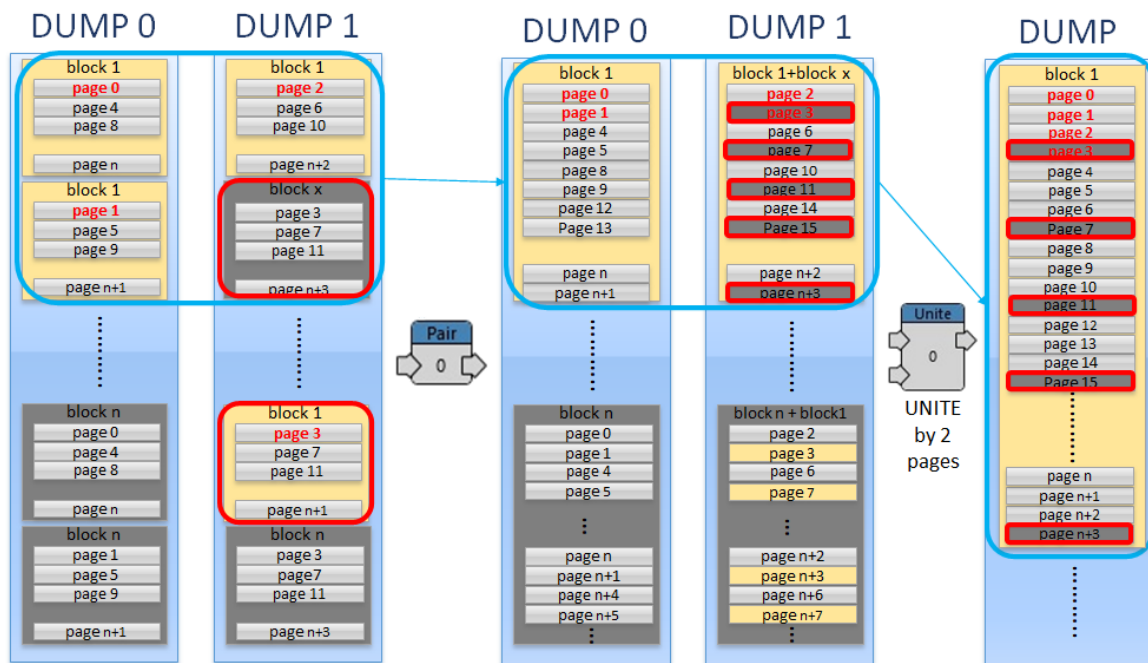
Asymmetric Multi-Plane Allocation

Asymmetric Multi-Plane page allocation is working in a similar way- one logical block is built from physical blocks from all planes. Usually, 80-90% of blocks are allocated in the standard way, but the rest may be built from physical blocks located by different offsets in the planes. And if such a block will be reconstructed in the standard way, it will be wrong, data will not be available there. The worst thing is that usually the file system blocks are written in the asymmetric manner, so the standard approach does not allow to get the file structure.

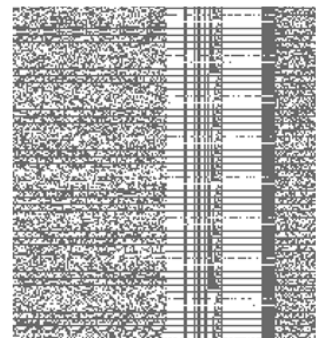
ASYMMETRIC MULTI-PLANE PAGE ALLOCATION ON THE CHIP LEVEL



ASYMMETRIC MULTI-PLANE PAGE ALLOCATION ON THE CHIP LEVEL



1 logical block =
4 physical blocks
In cases with
asymmetric page
allocation, some
of blocks could
be damaged.



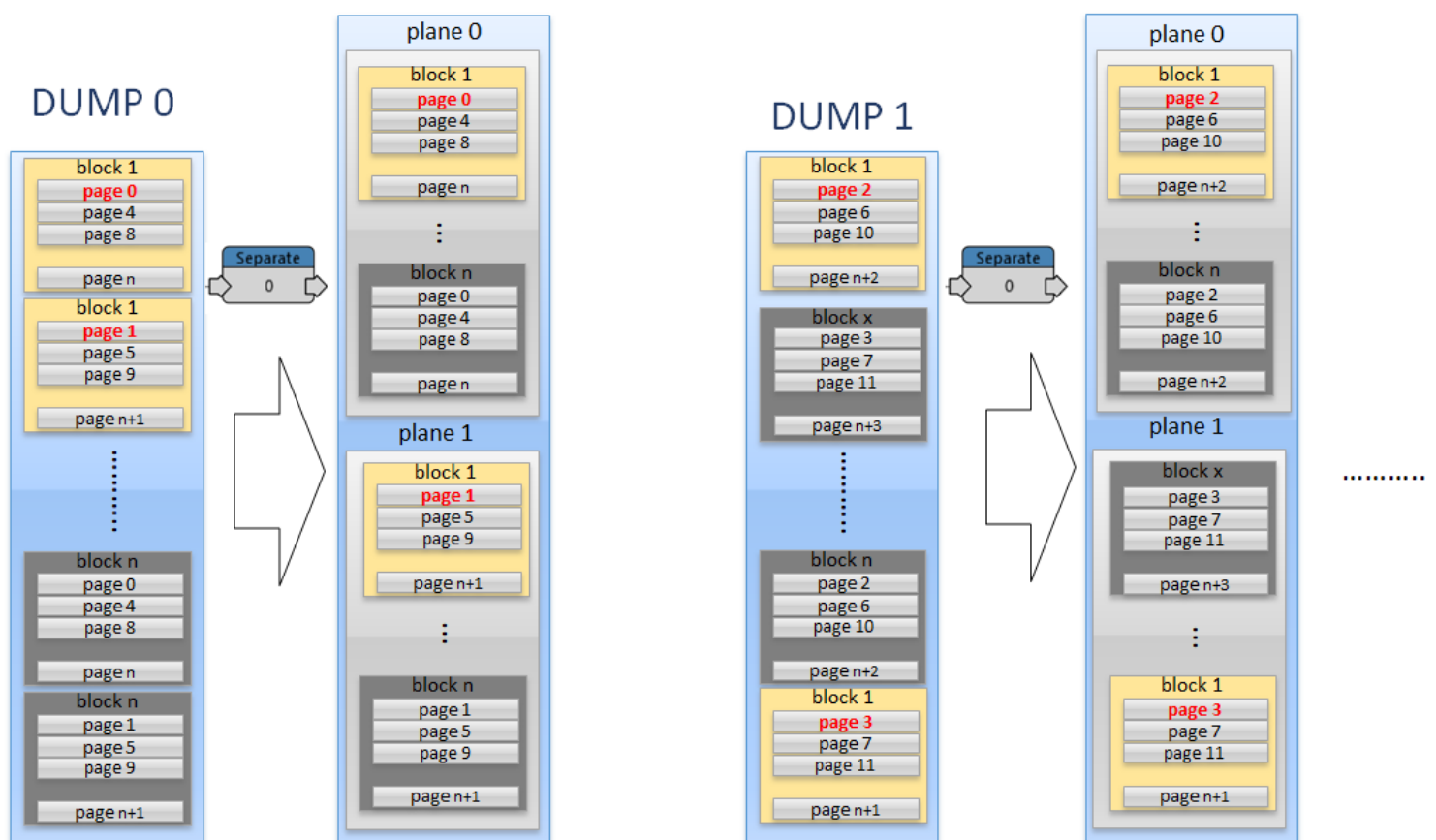
As it is shown, it corrupts data.

The good news is that VNR can rebuild the full image in the correct way,
the step-by-step instruction is below.

Let's say that some device has two crystals with two planes each. Logical blocks are written simultaneously into all 4 planes. But logical BLOCK 1 is written asymmetrically: the first 3 physical blocks are written by the same offset in planes and the last one is written by a different offset in the last fourth plane.

The dump from each crystal is one single file that consists of blocks from two planes. The blocks from Plane 0 are blocks with numbers 0,2,4,6,8... and so on in the dump, and the blocks from Plane 1 are blocks with numbers 1,3,5,7... in the dump.

The first step is to separate each dump by planes. The “Separate” element should be used for this. This element will place all blocks from PLANE 0 in the first half of the dump, and the blocks from PLANE 1 will be placed in the second half of the dump. The picture below shows how it works:



So now all blocks from each plane are grouped together.

In the next step, it is necessary to unite dumps with the UNITE element. So we gonna get a single dump with 4 parts. Each quarter contains physical blocks from minimal to maximal LBN from the corresponding plane.

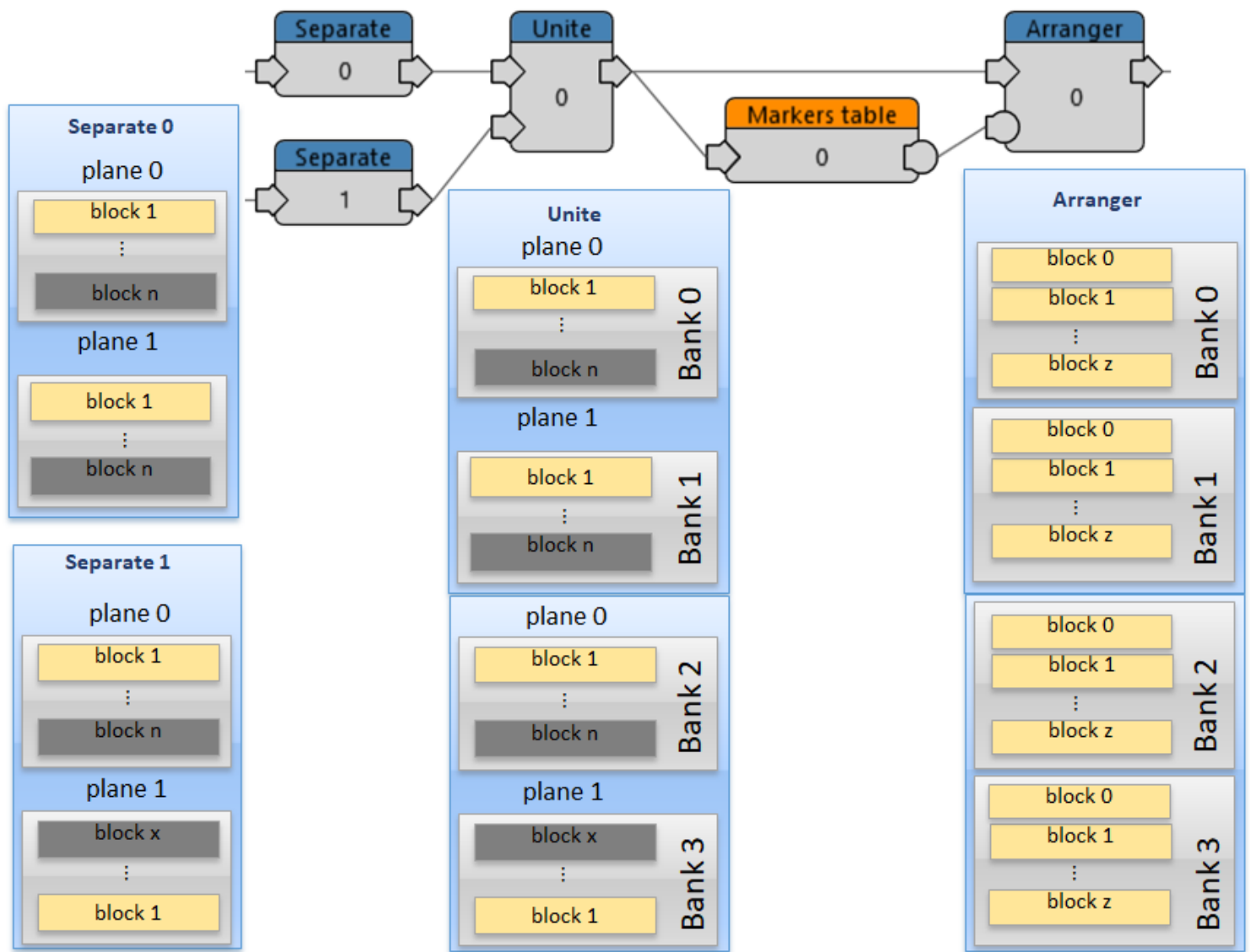
After that, the Markers table with 4 banks will be used to arrange blocks in ascending order. 4 banks should be used to keep grouping of the blocks by planes. After this there will be a dump where the first quarter is taken by blocks from the first plane and all of them are arranged in ascending order, the second quarter is taken by the arranged blocks from the second plane, and so on.

Of course it is essential to add the bank number to the Markers table according to the number of planes that are inside the UNITE element, so if we unite two dumps with two planes, the number of banks is 4, for two dumps with 4 planes the number of banks is 8, and so on...

The Markers table has the standard operations except one thing that is necessary to watch out.

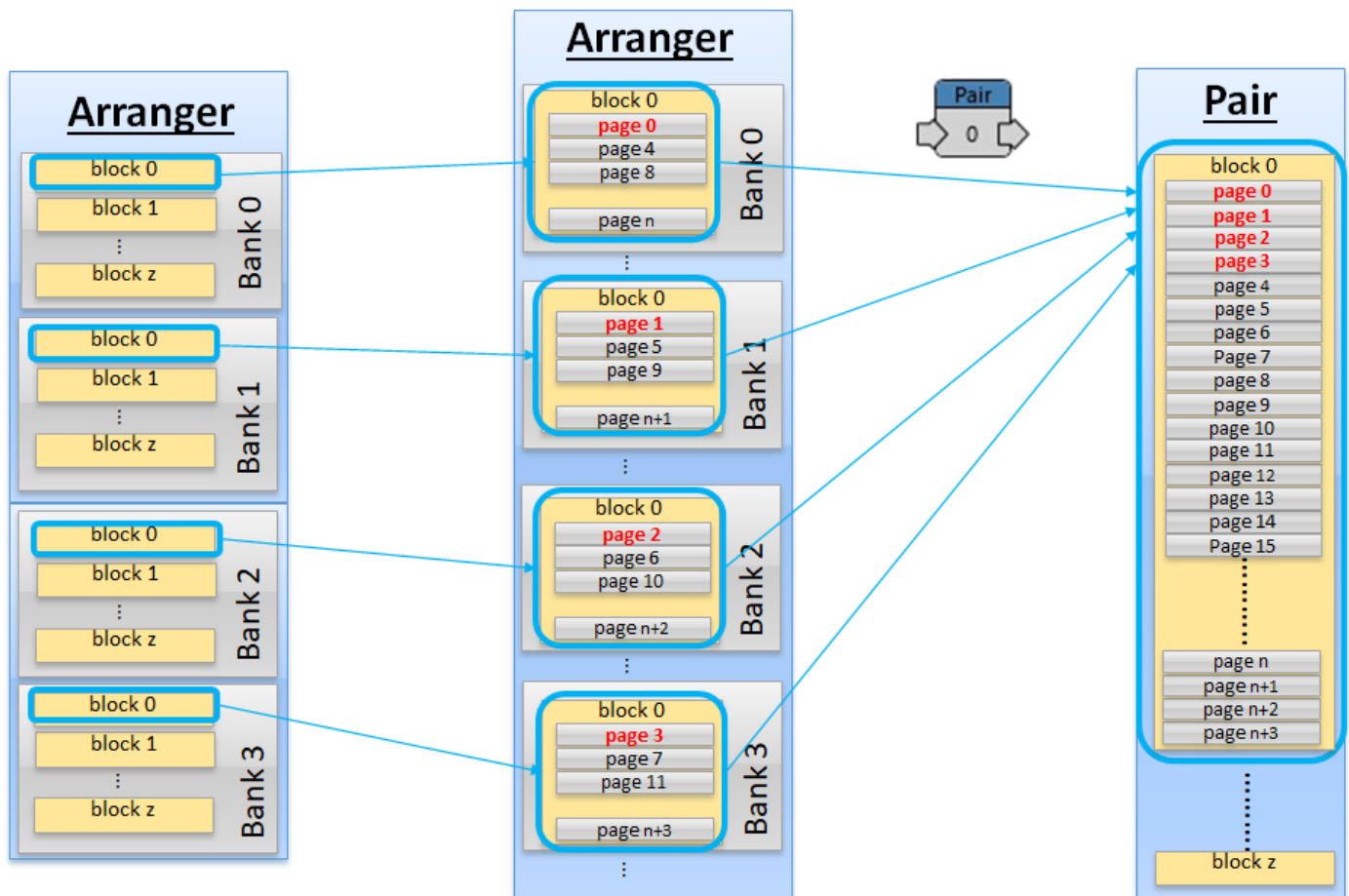
All Banks have to have the same number of blocks.

After sorting, selecting, and inserting missing blocks in the Markers table, the Arranger element is in use.



Ok, so now there is a dump where 4 banks with blocks are allocated sequentially in each. So there is Block 0 in the first position in the first Bank, Block 0 in the first position in the second Bank, and so on. And if you remember Multi-Plane page allocation theory, the pages were written simultaneously into all those 4 blocks. Now the task is to gather pages from 4 Blocks from 4 quarters to make PAGES go sequentially. The PAIR element will be used for this.

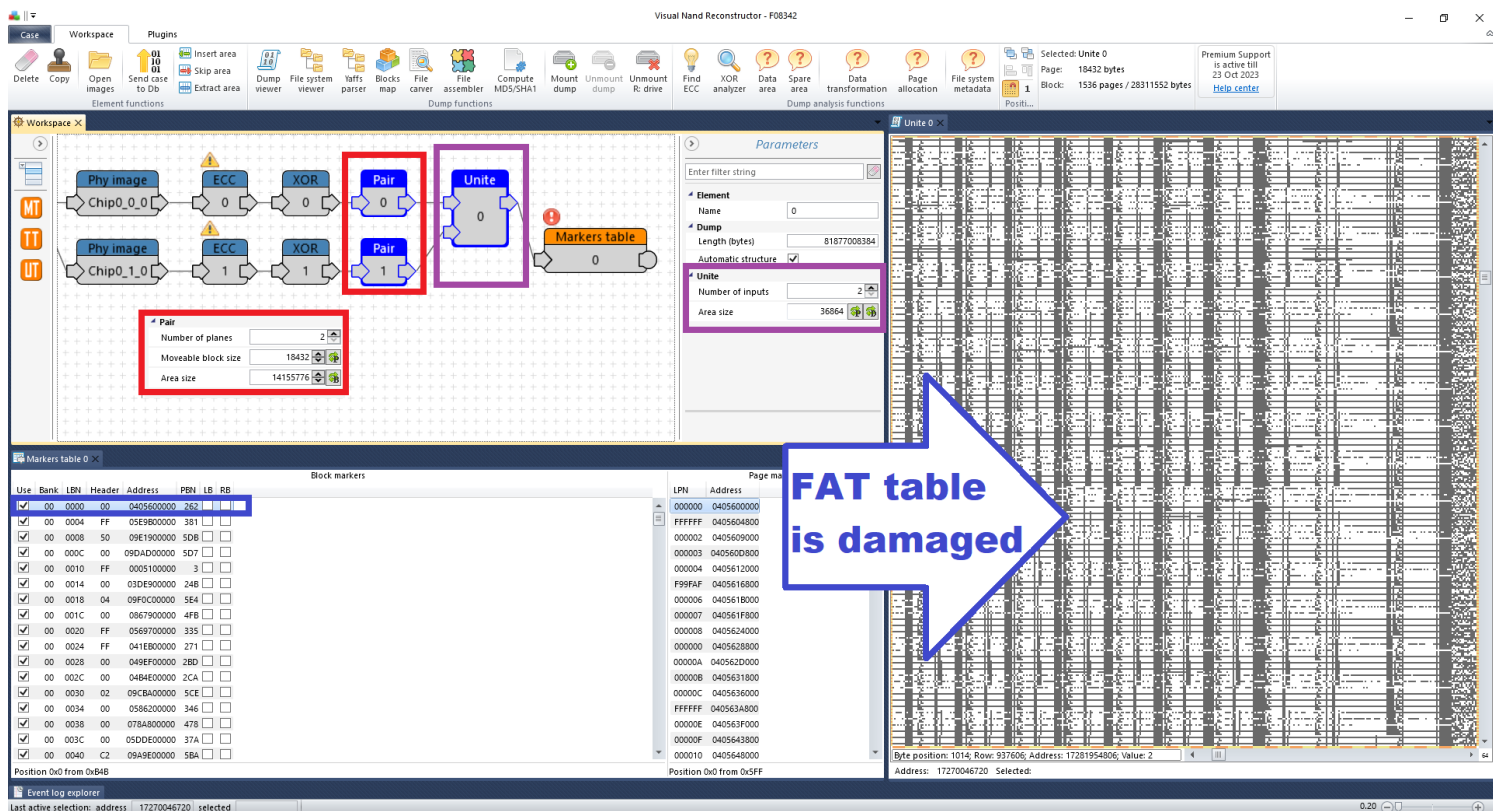
In the Pair element settings, the number of planes is equal to the number of banks in the Markers table, Moveable block size is equal to the page size, and Area size is the size of the Arranger element. With these settings, the PAIR element will take one Page from one quarter, the next Page from the next quarter, and so on, and then put it into the final image.



After those steps all blocks and pages inside them are in the correct order, the last step is to use the Data Area element to extract the full Logical Image.

Example on a real device:

On the first look, it is a standard PS-2251-07-V with multi-plane allocation, but after the PAIR and UNITE the first block with FAT table is damaged. In the bitmap, we can see that every 4th page is damaged. The file structure is not available now, but this pattern of damage is a clue for asymmetric multiplane allocation.



The first step to solve this case is to separate planes inside the dumps. This element is used after the XOR. After this, the separated planes should be united with the UNITE element by dumps.

Separate

Number of partition

2

Moveable block size

7077888

Area size

40938504192

SEPARATE SETS

Unite

Number of inputs

2

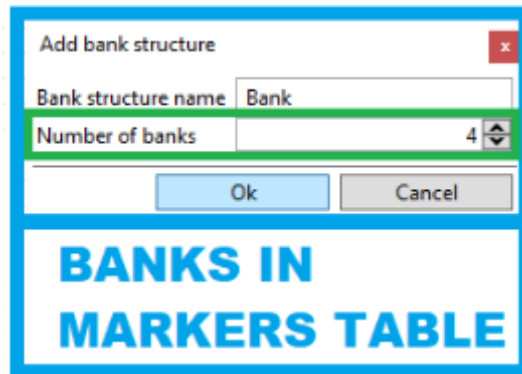
Area size

40938504192

UNITE SETTINGS

Then it is necessary to create a marker table, and the essential here is the bank number.

In this case, there are two chips with two planes each, so the number of banks is four.



The most important step is inside the marker table. If the number of blocks in one bank will be different this will damage the whole image. In this particular case, each bank has a range of blocks from 0000 to 9894, and the LBN step is 4/4 with "FCFC" mask, all banks have the same number of blocks.

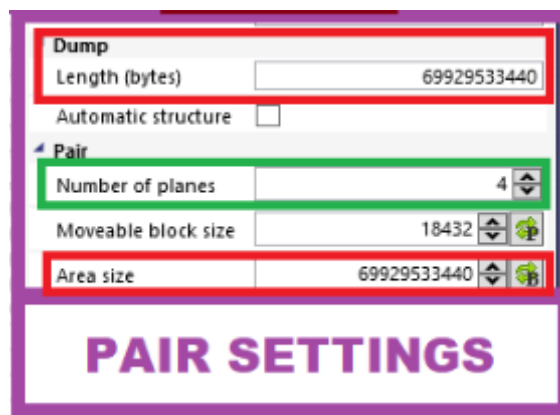
Use	Bank	LBN	Header	Test 1	Test 2	Address	PBN	LB	RB
☒	00	0000	00	02FF	55AA	0101580000	262	☐	☐
☒	00	0004	00	03FF	0000	00114C0000	29	☐	☐
: Bank 0									
☒	00	9890	00	02FF	758A	04BE940000	B3F	☐	☐
☒	00	9894	00	01FF	C0C2	03F2640000	95B	☐	☐
☒	01	0000	00	02FF	0000	04DFE80000	B8E	☐	☐
☒	01	0004	00	03FF	0000	063E7C0000	ECD	☐	☐
: Bank 1									
☒	01	9890	00	02FF	FC00	0982A40000	168B	☐	☐
☒	01	9894	00	01FF	0A26	08B6740000	14A7	☐	☐
☒	02	0000	00	02FF	FFFF	0A89780000	18FA	☐	☐
☒	02	0004	00	03FF	0200	0B028C0000	1A19	☐	☐
: Bank 2									
☒	02	9890	00	02FF	3808	0E46B40000	21D7	☐	☐
☒	02	9894	00	01FF	DE4C	0D7A840000	1FF3	☐	☐
☒	03	0000	00	02FF	5F5F	0F4D880000	2446	☐	☐
☒	03	0004	00	03FF	0000	0FC69C0000	2565	☐	☐
: Bank 3									
☒	03	988C	00	03FF	BD3F	127E480000	2BD6	☐	☐
☒	03	9890	00	02FF	52FE	130AC40000	2D23	☐	☐
☒	03	9894	00	01FF	AC3F	123E940000	2B3F	☐	☐
☐	00	0000	00	1000	CB19	0003CC0000	9	☐	☐
☐	00	0000	00	8000	C339	0004380000	A	☐	☐
: MARKERS TABLE INSIDE									

After selecting blocks it is necessary to click “create logical image” on the markers table element, but instead of the Logical image it is necessary to use the Arranger element, because further transformations are required.

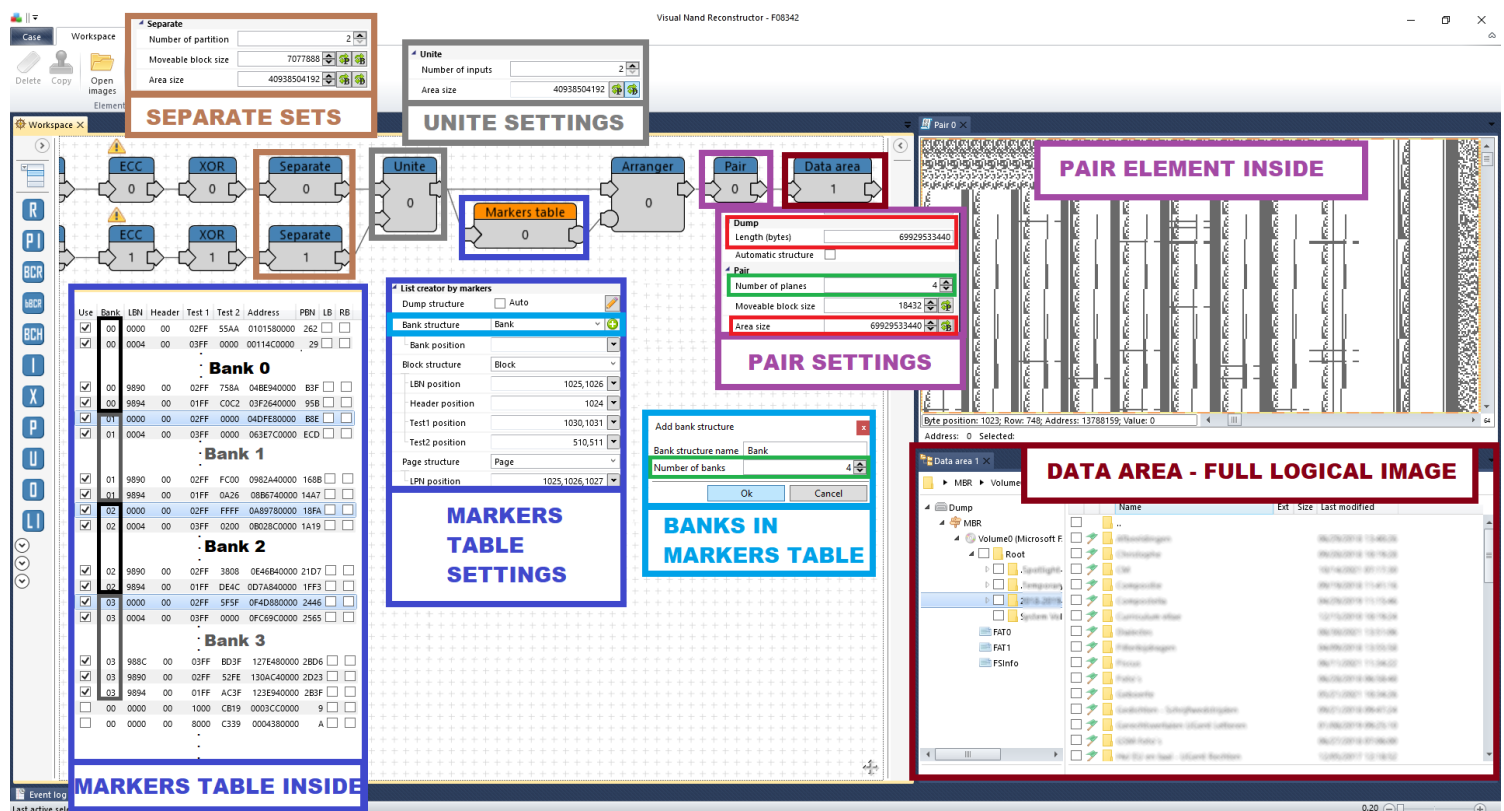
When all blocks are extracted in the Arranger element, the Pair should be connected to it.

Settings of the Pair element, are simple- the Moveable block size is defined automatically (this is page size), the number of planes is the same as the number of banks of the Marker table.

The area size is the size of the whole dump.



After the Pair, the Data area element should be connected in order to extract the data area. Now it has is fully correct logical image and the file system is available:



The number of asymmetric blocks is below 10% in devices with this issue, but if at least one of the first blocks is allocated in the asymmetric way, then the full dump with the file system is unavailable in a standard way, but as it is shown, with the VNR it is possible to solve it.